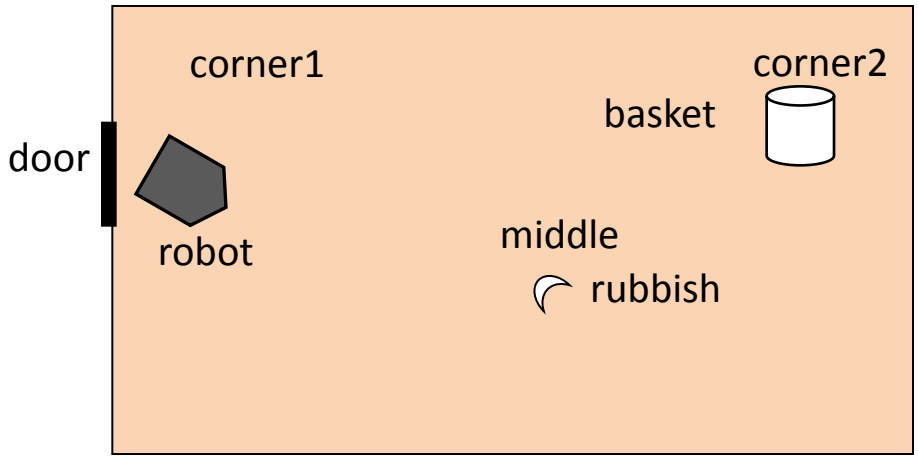
AN EXAMPLE PROGRAM: ROBOT TASK PLANNING

Ivan Bratko

University of Ljubljana

These slides are meant to be used with a Prolog system to demonstrate the examples, and the book: I. Bratko, Prolog Programming for Artificial Intelligence, 4th edn., Pearson Education 2011. The slides are not self-sufficient.

A CLEANING ROBOT IN A ROOM



STATES OF THE ROBOT'S WORLD AND ACTIONS

% State of the robot's world =

% state(RobotLocation, BasketLocation, RubbishLocation)

% action(State, Action, NewState): Action in State produces NewState

% We assume robot never drops rubbish to floor,

% and never pushes rubbish around

```
action(   state( Pos1, Pos2, floor(Pos1)),     % Robot and rubbish at Pos1
          pickup,                             % Pick up rubbish from floor
          state( Pos1, Pos2, held) ).         % Rubbish now held by robot
```

```
action(  state( Pos, Pos, held),          % Robot and basket both at Pos
         drop,                            % Drop rubbish to basket
         state( Pos, Pos, in_basket)).    % Rubbish now in basket

action(  state( Pos, Pos, Loc),           % Robot and basket both at Pos
         push( Pos, NewPos),              % Push basket to NewPos
         state( NewPos, NewPos, Loc)).    % Robot and basket at NewPos

action(   state( Pos1, Pos2, Loc),
         go( Pos1, NewPos1),              % Go from Pos1 to NewPos1
         state( NewPos1, Pos2, Loc)).
```

```
% plan( StartState, FinalState, Plan):
%    Plan is a list of actions that trasnform StartState into FinalState

plan( State, State, [ ]).              % Goal state already reached

plan( State1, GoalState, [ Action1 | RestOfPlan])  :-
   action( State1, Action1, State2),              % Make first action
   plan( State2, GoalState, RestOfPlan).          % Find rest of plan
```

PLAN TO PUT BANANA PEAL INTO BASKET

?- plan(state(door, corner2, floor(middle)), state(_, _, in_basket), Plan).

Plan = [go(door,middle), pickup, go(middle,corner2), drop]

- Study *how* our program found this plan (trace program's execution)

- Let Prolog find alternative plans:

?- plan(state(door, corner2, floor(middle)), state(_, _, in_basket), Plan).

Plan = [go(door, middle), pickup, go(middle, corner2), drop] ;

Plan = [go(door, middle), pickup, go(middle, corner2), drop, push(corner2, _A)] ;

Plan = [go(door, middle), pickup, go(middle, corner2), drop, push(corner2, _A),

push(_A, _B)] ;

...

- Prolog keeps expanding the initial plan with pushing the basket back and forth
- Why doesn't Prolog produce alternative shorter plans?

FORCE THE PLANNER TO SEARCH IN BREADTH-FIRST FASHION

- An easy way is to limit the length of plans, and gradually increase the length limit
- Idea: **conc** can be asked to generate general lists of increasing length:

?- conc(L, _, _).

L = [ ];

L = [_A]; % _A is a a Prolog-generated name of a variable

L = [_A, _B]; % _A and _B are Prolog_generated variables

...

FORCE THE PLANNER TO SEARCH IN BREADTH-FIRST FASHION, CTD.

?- conc(Plan _, _), % Generate plan templates, short first

plan(state(door, corner2, floor(middle)), state(_, _, in_basket), Plan).

Plan = [go(door,middle), pickup, go(middle,corner2), drop] ;

Plan = [go(door,corner2), push(corner2,middle), pickup, drop)] ;

Plan = [go(door,middle), pickup, go(middle,corner2), drop, push(corner2, _A)] ;

...